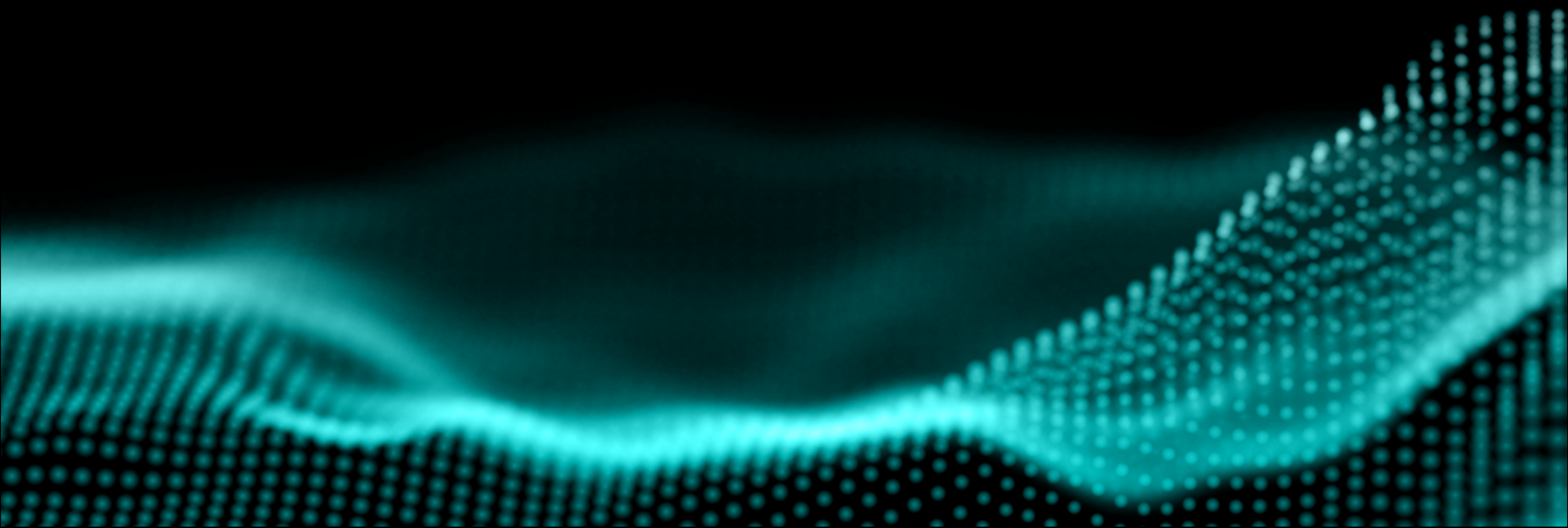
UNDERSTANDING OBSERVABILITY PIPELINES 02

Cribl

WHITE PAPER

Why Log Systems Require So Much Infrastructure *(and Three Ways to Fix Them)*

BY CLINT SHARP, CO-FOUNDER & CEO, CRIBL



Cribl

WHITE PAPER

Why Log Systems Require So Much Infrastructure *(and Three Ways to Fix Them)*

by Clint Sharp, Co-Founder & CEO, Cribl

Overview

Log systems like Splunk or ElasticSearch, by the standards of most data analytics systems, are easy to get data into and query. Compared to traditional databases or data warehouses, log systems generally require very little planning. This success has led to a new problem, where log systems have become the dumping ground for many use cases for which they are poorly suited.

Log systems are pulling triple duty across many enterprises today. The first workload is placing unstructured data into an index to provide fast search for troubleshooting and investigation. For this use case, they provide very high value; if you need to be able to dive through huge piles of machine data quickly, they are by far the best tool for the job.

In addition to providing fast needle-in-a-haystack event search, log systems have become the de facto place to store all log data, whether it's being used for investigations or whether it's being collected for compliance or just-in-case. In many use cases, especially for compliance, log data simply needs to be stored, but is rarely, if ever queried. Security investigation teams would like to retain years of high fidelity data so breach investigations can get a view of what was happening months or years in the past. Unfortunately, if you use a log system to store bulk data, you also have to index the data. In these use cases, optimizing for fast search carries significant costs which grow linearly with retention time.

OPTIMIZING FOR FAST SEARCH CARRIES SIGNIFICANT COSTS WHICH GROW LINEARLY WITH RETENTION TIME.

Additionally, log systems are used as systems of analysis for large data sets like flow logs or web access logs. Large time series datasets like this are often aggregated and summarized, but rarely do users go look for one individual record in the data set. These datasets can be analyzed in the stream to glean insight without the need to index every log event for fast search. This data is better stored as metrics in a time series database or as summaries in the logging system. Doing this requires knowing in advance what questions you want to ask of your data, but paired with storing full fidelity raw information in a more affordable location, we aren't discarding information.

STORING DATA IN OBJECT STORAGE COULD BE AS LITTLE AS 1% OF THE COST OF STORING IT ONLINE IN AN INDEXING ENGINE.

Indexing data for fast retrieval is *expensive*! Indexing requires significant computing power in your ingestion pipeline to create the indexes. The raw data is compressed, usually at about 10:1, but indexing usually requires 3x to 4x the size of the compressed raw data. Solely indexing the data uses ~40% of the storage size of the raw data ingested.

Then, in order to provide resiliency, the data is replicated usually an additional 2 times, meaning we're storing 120% of the size of the raw data. In addition, searching an index requires fast disk, so log systems require expensive block storage on the storage nodes rather than cheap object storage.

Storing data as compressed files in an object store requires 10% or less of the storage required for indexing. Object storage can be as little as 12.5% of the cost of block storage. Combined, storing data in object storage could be as little as 1% of the cost of storing it online in an indexing engine.

Analyzing the data in the stream and providing aggregated summary statistics provides the data to drive the organization's dashboards – while providing massive savings relative to just indiscriminately indexing all your data. Having an online archive in object storage, of full fidelity data that can be easily retrieved, means nothing is ever lost, as the data can be replayed later.

The remainder of this whitepaper will walk through:

- *Why we index data to begin with online in an indexing engine.*
- *When that optimization becomes wasteful*
- *Why object storage is so much cheaper*
- *Where it can be put to use*

We'll end with our recommendations for how best to build a cost effective log data management for the future.

Why Indexes Exist

If you're just becoming familiar with the problem, it would be fair to ask why we even index data to begin with, if it's so expensive. Indexing log data was a huge innovation pioneered by Splunk in the mid-'00s. Historically, log analysis was the domain of *grep*, the Unix command line utility that chewed through raw text files and only spit out lines that match a given search string. You can think of *grep* as like trying to find every instance of a particular word in a book by looking at every page, and looking for a match while keeping a record of every page that matched. It's a slow process. As log volumes grew into the gigabytes and terabytes at rest, scanning through raw log data to find a rare search string took an incredibly long time. Investigations thrive on being able to quickly explore hypotheses, which is really hindered by every question taking minutes to ask.

[INDEXING] HAS BEEN STRETCHED TO BE A ONE SIZE FITS ALL SOLUTION FOR ALL LOG DATA PROBLEMS.

The inspiration for solving this problem was found in Internet search systems. Everyone likely uses Google or another search engine dozens of times per day. Splunk, and later other systems like Elasticsearch, began treating log events like a document to be searched and creating indexes for faster retrieval of the matching events.Implementation details had to be modified to deal with very large numbers of small documents, but the same principles apply. This is akin to trying to find all instances of a word in a book by flipping to the index at the back of the book first, and then navigating in the book to each instance of that word. Using the index is

ADMINISTRATORS CAN ADOPT A DISCERNING DATA MANAGEMENT STRATEGY USING DATA STORAGE TECHNIQUES WHICH ARE FIT FOR PURPOSE.

way faster than reading every page of a book trying to find all instances of the word.

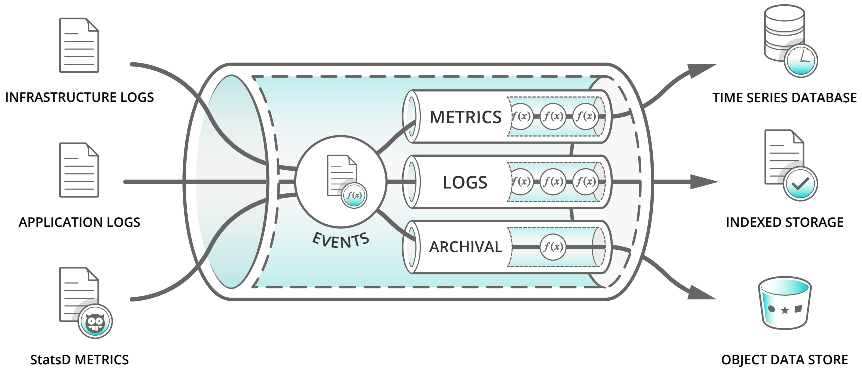
Indexes require additional storage in addition to the raw text. Indexes can greatly reduce the amount of time and computing required to go find rare terms in large data sets.

When Indexes are Wasteful

The ability to rapidly find rare terms in terabytes or petabytes of raw data is a massive innovation. It's such a successful approach that multiple billion dollar companies have sprung up from this approach. Unfortunately, as is often the case, this core innovation has been stretched to be a one size fits all solution for all log data problems. For some workloads, indexes are a very wasteful optimization.

Back to our real world analogy of a book, what if I never need to find instances of words in the text? Rarely do you find an index at the end of a fiction book; it would be wasteful to generate an index and use the additional paper when nobody is going to use it. And what if you wanted to find out how many times *the* appeared on every page of the book? Assuming you had an index, *the* would likely appear on every page, so going to the back of the book, seeing that it indeed was included on the next page, and then scanning back to the next page would be much slower than just going page by page and counting the number of instances of *the*.

Computing workloads are actually pretty directly analogous to a human doing these same types of analysis in a book. If we're never going to use the data, indexes are wasteful. If we're just going to have to read every log event, like bulk analytical or aggregation queries, indexes are slower than just scanning linearly.



Online Storage Cost

In order to perform well, indexing engines need to have indexes and raw data collocated on fast disk. Similar to the real world operation we described earlier in a physical book, querying an index is an exercise in jumping around in a data set.

First, you examine the postings table to find all the documents that match your given query, then you scan the raw data to retrieve the appropriate records. This requires disk that allows random access, and it is difficult to do on partial datasets. Traditional logging systems like Splunk Enterprise and Elasticsearch manage these datasets for you in a clustered approach.

They'll ensure the data is replicated to minimize the risk of data loss. In these approaches, all data that can be queried must be directly online. Approaches like Splunk Smart Store allow for separation of storage and compute by treating the compute layer as an ephemeral cache, which reduces the amount of block storage required, but you still need fast disk for however much data you'd like to keep in the cache.

Implementers of these current systems have to opt for much more expensive block storage, and usually the even faster IOPS guaranteed storage. For archival storage, or in order to maintain an online data lake which contains the raw data for reprocessing, we can use object storage at significantly more cost effective rates.

Affordable System of Record for Logs

Instead of treating log indexing engines as one size fits all data stores, administrators can instead adopt a more discerning data management strategy which utilizes a number of data storage techniques which are fit for purpose. Looking at this visually, for our three use cases, we can choose three data management strategies.
As we've examined in this article, we've modified the original one size fits all indexing engine architecture to now include an observability pipeline which splits the data into three different destinations.

1. *The* **traditional log indexing engine** *is where we put data which benefits from needle in a haystack search performance.*

2. *The* **time series database** *is where we place metrics data created by running aggregate statistics and sampling in our pipeline. The TSDB provides fast dashboarding and initial investigation.*

3. *The third destination is our* **system of record** *for all of this data. Here, we place raw data in cheap storage, well partitioned, for optimized retrieval of subsets of our raw data. This data can be replayed back to any indexing engine, TSDB, or analytics database for analysis. By implementing this strategy, we can often save 50% or more in the total cost of a solution for logging, both for observability and security.*

If this is interesting to you, Cribl LogStream™ implements an observability pipeline which allows you to create an affordable system of record for logs. Please check out our Sandbox which shows how we can connect legacy agents easily to S3 storage. Our upcoming release, 2.2, will offer Ad-Hoc data collection making it easy to replay data from object storage or a filesystem.

ABOUT CRIBL

Cribl is a company built to solve customer data challenges and enable customer choice. Our solutions deliver innovative and customizable controls to route security and machine data where it has the most value. We call this an observability pipeline, and it helps slash costs, improve performance, and get the right data, to the right destinations, in the right formats, at the right time. Join the dozens of early adopters, including market leaders such as TransUnion and Autodesk, to take control and shape your data. Founded in 2017, Cribl is headquartered in San Francisco, CA. For more information, visit www.cribl.io or our LinkedIn, Twitter, or Slack community.